REMARKS

Claims 1, 11, 14 and 25 have been amended for purposes of clarity only and thus for reasons unrelated to patentability. Support for the amendment of Claims 1, 11, 25 appears in the specification at least at: page 9, line 30 to page 10, line 10; page 14, lines 7-29; page 16, line 23 to page 17, line 14; and in FIGS. 3, 4, 5. Claims 21-24 have been canceled without prejudice.

The headings below are numbered to correspond with the heading numbering used by the Examiner in the Office Action.

Request for Examiner Interview.

Should the Examiner be of the opinion that this Amendment does not place the application in a condition for allowance, Applicant respectfully requests an Examiner Interview prior to the issuance of the next communication from the USPTO to expedite prosecution.

1-3. Restriction Requirement.

Applicant hereby affirms the election of invention I, Claims 1-20, 25. Accordingly, Claims 21-24 have been canceled without prejudice.

4. Claim 25 satisfies 35 U.S.C. § 101.

The Examiner states:

> Applicant's disclosure … provides intrinsic evidence that would lead one of ordinary skill to reasonably interpret the claimed computer program product as a signal transmitted over a network. (Office Action, page 3.)

Claim 25 has been amended and now recites:

> A computer-program product comprising a **tangible** computer-readable medium containing computer program code comprising… (Emphasis added.)

GUNNISON, McKAY &
HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Page 6 of 12

Accordingly, Claim 25 satisfies 35 U.S.C. § 101.

For the above reason, Applicant respectfully requests reconsideration and withdrawal of this rejection.

5-6.   Claim 14 satisfies 35 U.S.C. § 112, second paragraph.

The Examiner states:

> … it is unclear what is meant by "determining said block" such that the scope of the claims is indefinite. (Office Action, page 3.)

Claim 14 has been amended and now recites:

> The method of Claim 11 further comprising determining **said block being released by said application**. (Emphasis added.)

It is noted that Applicant's specification in reference to FIG. 6 sets forth:

> In DETERMINE BLOCK DEALLOCATED OPERATION 610, the block of the heap buffer released by the program is determined. In one embodiment, parameters associated with the heap deallocation function call are analyzed to determine the deallocated block. (Page 21, lines 2-6.)

Applicant respectfully submits that one of skill in the art would understand what is being claimed in Claim 14 when read in light of the specification. Accordingly, Claim 14 satisfies 35 U.S.C. § 112, second paragraph.

For the above reasons, Applicant respectfully requests reconsideration and withdrawal of this rejection.

7-9.   Claims 1-20, 25 are patentable over Fetzer et al. (6,832,302) in view of Abrashkevich et al. (2004/0221120).

The Examiner admits:

> **Fetzer does not explicitly disclose**: Determining if a forward link (F-link) and a backward link (B-link) of said predicted block are addresses within a heap segment

associated with said predicted block … (Office Action, page 4, emphasis added.)

To cure this glaring deficiency in Fetzer et al., the Examiner further asserts:

> … Fetzer suggests this limitation by disclosing that the **buffer space may be kept track of** by wrapping allocation functions (such as malloc, calloc, free, etc.) and keeping meta-data for each allocated buffer (*Fetzer-Col 2 lines 34-38*). Furthermore Fetzer discloses that strcpy fault-containment wrapper 131 may determine if destination start address 163 is part of any allocated buffer in heap 160. **If the destination start address does not fall within a heap buffer**, then an error handling procedure is executed (*Fetzer-Col 6 lines 34-37*). (Office Action, page 4, emphasis added.)

Applicant notes that Fetzer et al. teaches that the method of detecting heap smashing occurs when data is written to the heap. The Examiner has failed to callout where Fetzer et al. teaches or suggests that **values already within the heap are analyzed**.

More particularly, the method of Fetzer et al. includes two operations.

First, as noted by the Examiner, a determination is made as to whether the destination start address **where the data is to be written** falls within a currently allocated heap buffer. Specifically, Fetzer et al. teaches:

> If **the data is being written** to the heap, the fault-containment wrapper may determine whether copying the string to the destination would have caused the heap to be smashed and might take appropriate action based on the conclusion of this determination. In one embodiment, the fault-containment wrapper **determines whether the destination start address falls within a currently allocated heap buffer** (305). … If the destination start address does not fall within a heap buffer, then an error handling procedure is executed **instead of writing the data to the heap** (306) (Col. 6, lines 26-38, emphasis added.)

Second, if the destination start address is within a currently allocated heap buffer, then a determination is made as to whether **writing the data** would overflow the buffer. Specifically, Fetzer et al. teaches:

> If the destination start address does fall within a the heap the buffer, then the fault-containment wrapper determines **the size of the data-block to be written to the heap** (306) and then determines **if the data-block's size is greater than the size of the memory section which starts at the destination start address and ends at the end of the buffer** (308). That is, the fault-containment wrapper determines if writing the data would overflow the buffer. … If **writing the data** would overflow the buffer (as shown in FIG. 2), then the error handling procedure is performed (307). (Col. 6, lines 42-56, emphasis added.)

In Summary, Fetzer et al. teaches:

> … the fault-containment wrapper may be used to check whether **a destination start address is within the boundary of a buffer** and, if so, **whether the string being written starting at that destination start address will fit within the buffer.** (Col. 7, lines 5-9, emphasis added.)

To determine whether the destination start address is within **the boundary of a buffer** and, if so, whether the string being written starting at that destination start address will fit **within the buffer**, Fetzer et al. of course teaches that the **"buffer space may be kept track of"** as stated by the Examiner. However, as demonstrated above, the Examiner has failed to callout where Fetzer et al. teaches or suggests that **values already within the heap are analyzed.**

Abrashkevich et al. fails to cure this deficiency in Fetzer et al. Regarding Abrashkevich et al., the Examiner states:

> … Abrashkevich teaches defensive heap memory management … wherein doubly linked lists are implemented (including an f-link and a b-link) as memory management mechanisms to manage and keep track of allocated and free blocks/chunks of memory (*Abrashkevich-paragraphs*

0003, 0025, first ~ 17 lines of paragraph 0026, 0029).
(Office Action, page 5.)

Initially, Applicant notes that Abrashkevich et al. teaches
that the **linked lists are separate structures than the memory
chunks themselves.** Specifically, Abrashkevich et al. teaches:

> The defensive approach used in an embodiment of
> the invention is based on separating control
> information regarding chunk attributes such as, memory
> chunk size, its current status (free/in use), **links to
> the next and previous chunk,** etc. **from memory chunks
> themselves** and putting such control information into a
> relevant data structure within a protected (by special
> hidden walls described below) heap or pool header,
> **separating metadata from data.** … A preferred
> embodiment utilizes a randomized data structure called
> **a skip list** which uses extra links in the nodes of a
> linked list in order to skip portions of a list during
> a search. … Having separate lists for free and
> allocated memory chunks (pools or blocks) **and
> isolating the list control information** (by putting the
> lists of relevant memory chunks into a Heap Header 12
> or Pool Header 22 and Pool Header 24 as shown in FIG.
> 1) **from the rest of Heap Data 15 or Pool 14 or Pool 16
> space** allows management of a heap or pool even if some
> of the allocated chunks become corrupted. (Paragraphs
> [0025]-[0026], emphasis added.)

Accordingly, Abrashkevich et al. teaches that the linked
lists are metadata that is isolated from the memory chunks
themselves. Accordingly, the Examiner has failed to callout
where the memory chunks themselves contain a f-link or a b-link.

Further, Abrashkevich et al. teaches that attachments to the
memory chunks are analyzed to determine if the memory chunk is
corrupt. Again, the Examiner has failed to callout where a value
already within the memory chunk itself is analyzed.

Specifically, Abrashkevich et al. teaches:

> Referring to FIGS. 6a, 6b, and 6c in a preferred
> embodiment, each Allocation 382 has hidden 24 byte
> **Front MDIA 380,** as in FIG. 6a, and an optional **Back
> MDIA 384** as in FIG. 6b, **containing debugging
> information about the chunk and a special signature**

area which is used for verifying whether memory has been corrupted. (Paragraph [0069], emphasis added.)

For at least the above reasons, Fetzer et al. in view of Abrashkevich et al. does not teach or suggest:

> A method comprising:
> stalling a heap allocation function call to a heap allocation function originating from a request by an application for a block of heap buffer;
> predicting a predicted block of said heap buffer to fulfill said request, **said predicted block comprising** a header portion **and a data portion reserved for data;** and
> determining **if a forward link (F-link) in a F-link field and a backward link (B-link) in a B-link field** of said header portion of said predicted block **are addresses** within a heap segment associated with said predicted block,

as recited in amended Claim 1, emphasis added. Accordingly, Claim 1 is allowable over Fetzer et al. in view of Abrashkevich et al. Claims 2-10, which depend from Claim 1, are allowable for at least the same reasons as Claim 1.

Claims 11, 25 are allowable for reasons similar to Claim 1. Claims 12-20, which depend from Claim 11, are allowable for at least the same reasons as Claim 11.

For the above reasons, Applicant respectfully requests reconsideration and withdrawal of this rejection.
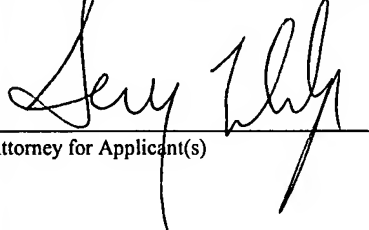
Conclusion

Claims 1-20, 25 are pending in the application. For the foregoing reasons, Applicant respectfully requests allowance of all pending claims. If the Examiner has any questions relating

Appl. No. 10/796,358
Amdt. dated August 21, 2007
Reply to Office Action of May 31, 2007

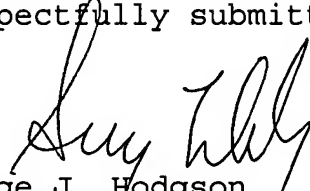to the above, the Examiner is respectfully requested to telephone the undersigned Attorney for Applicant(s).

Respectfully submitted,

Serge J. Hodgson
Attorney for Applicant(s)
Reg. No. 40,017
Tel.: (831) 655-0880